

FIG.1A

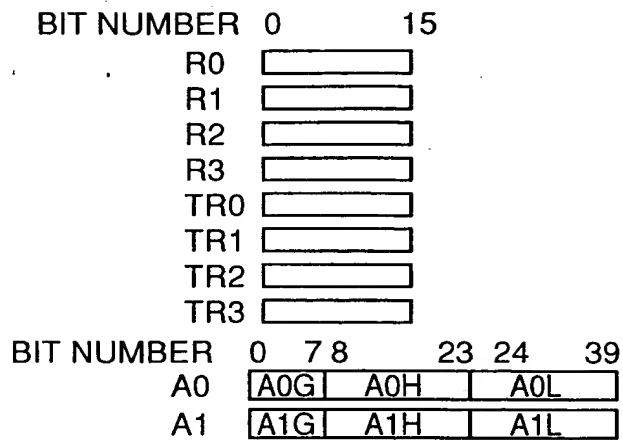


FIG.1B

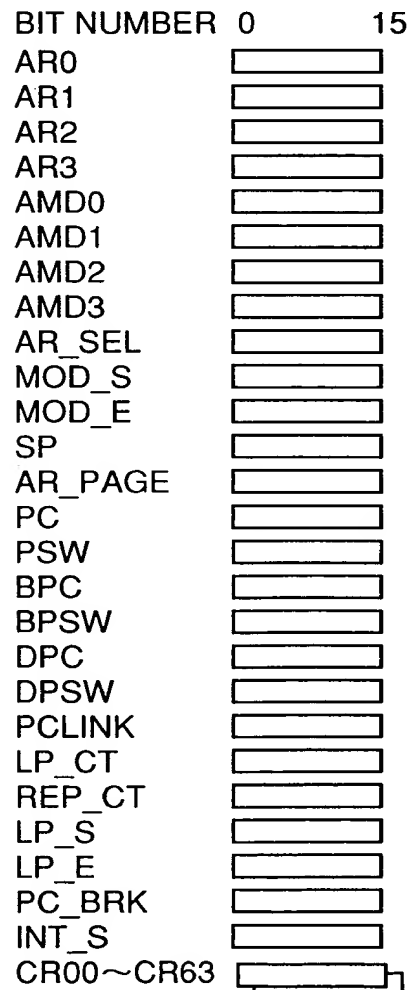


FIG. 2

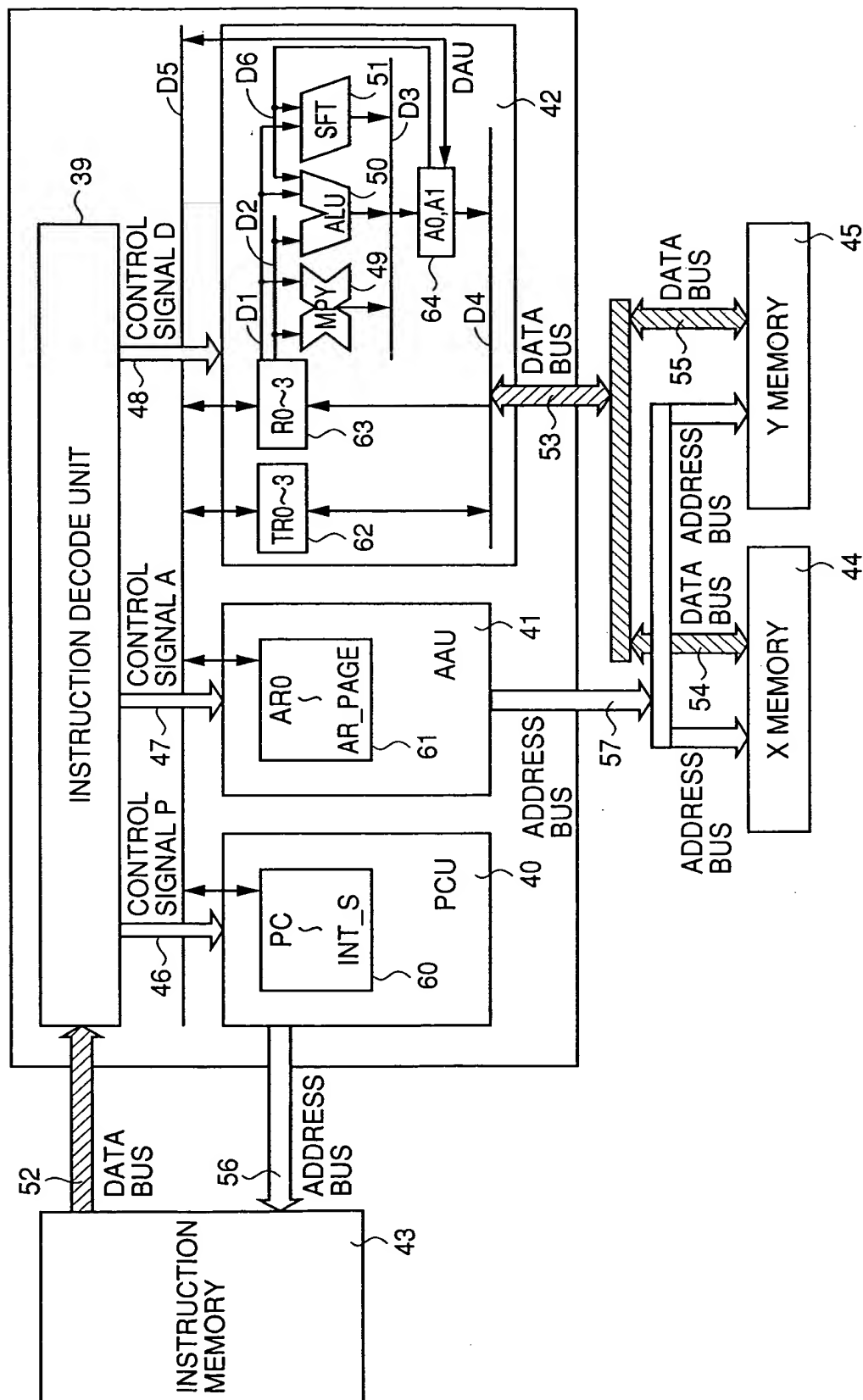


FIG.3

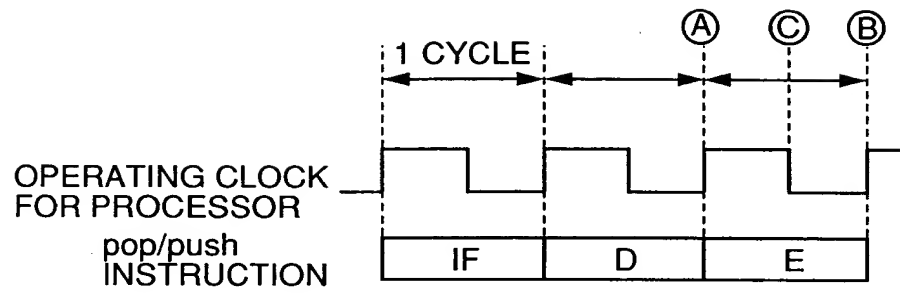


FIG. 3

FIG.4

OPERATION INSTRUCTION

mul	Multiply
muluu	Multiply unsigned operands
mac	Multiply and add
macuu	Multiply unsigned operands and add
macsu	Multiply signed operand by unsigned operand and add
macsul	Multiply signed operand by unsigned operand and add with shift right
macsuh	Multiply signed operand by unsigned operand and add with shift left
msub	Multiply and sub
msubuu	Multiply unsigned operands and sub
msubsul	Multiply signed operand by unsigned operand and add with shift right
msubsuh	Multiply signed operand by unsigned operand and add with shift left
add	Add a register to acc high
addl	Add a register to acc low
sub	Subtract a register from acc high
subl	Subtract a register from acc low
min	Set minimum value of acch or reg to accumulator
max	Set maximum value of accl or reg to accumulator
amin	Set minimum value to dest-acc
amax	Set maximum value to dest-acc
sra	Shift arithmetic right or left an accumulator
srl	Shift logical right or left an accumulator
and	And
or	Or
xor	Xor
nop	No operation
trfh	Transfer to an accumulator high
trfl	Transfer to an accumulator low
trf	Transfer to an accumulator
aadd	Add accumulators
asub	Subtract src-acc from dest-acc
sadd	Add dest-acc and src-acc with shift
abs	Absolute an accumulator
neg	Negate an accumulator
test	Test an accumulator(acc<0:set Nflag, acc==0:set Zflag)
md	Round an accumulator
not	Not an accumulator

**FIG.5**

**TRANSFER INSTRUCTION**

mv	Copy one word from a register to a register
ldi	Load immediate
ld	Load
st	Store
push	Push to stack
put	Put to stack
pop	Pop from stack

**SEQUENCE CONTROL INSTRUCTION**

jmp	Jump
call	Jump & link
loopi	Set loop counter and start hardware DO loop
loop	Start hardware DO loop
repeati	Set repeat counter and repeat next instruction
repeat	Repeat next instruction
return	Return from subroutine
reit	Return from EIT
rtd	Return from debugger EIT

**SPECIAL INSTRUCTION**

adr_set	Set AR_SEL register
mvin	Move from IO registers
mvout	Move to IO registers
slave	Transit to slav mode
noop	No operation

**FIG.6**

	ADDRESSABLE REGISTER
LOAD INSTRUCTION	R0, R1, R2, R3, TR0, TR1, TR2, TR3
STORE INSTRUCTION	TR0, TR1, TR2, TR3, A0H, A0L, A1H, A1L

**FIG.7** {

```

LDI AR3, #STACK_BOTTOM ; (1)
LDI AMD3, #DEC_1 ; (2)
. . . . .
ST TR0, X:AR3 ; (3)
MV TR0, AR0 ; (4)
ST TR0, X:AR3 ; (5)

```

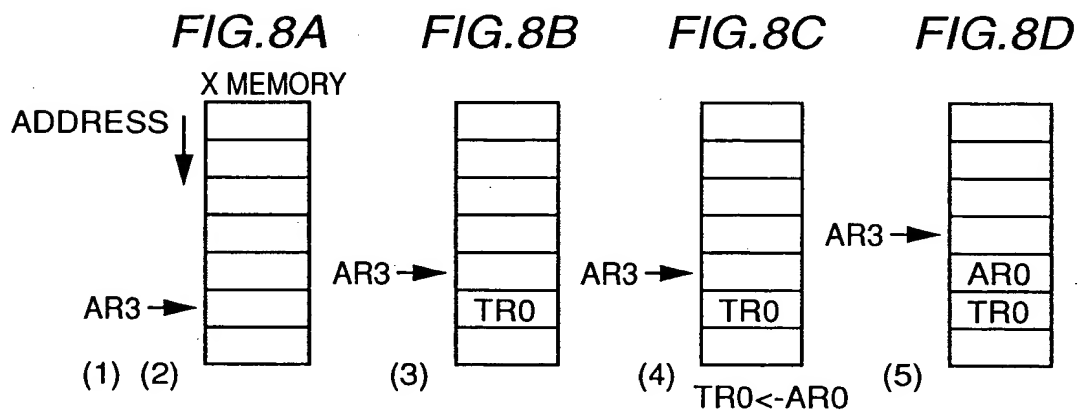


FIG. 9A

POP  
[mnemonics]  
    (1) pop  
    (2) pop ra  
[operation]  
    (1) tr0 = x\_memory[sp];  
        sp++;  
    (2) ra = tr0;  
        tr0 = x\_memory[sp];  
        sp++;

FIG. 9B

PUSH  
[mnemonics]  
    (1) push  
    (2) push ra  
[operation]  
    (1)  
        sp--;  
    (2) x\_memory[sp] = tr0;  
        tr0 = ra;  
        sp--;

FIG. 9C

PUT  
[mnemonics]  
    put  
[operation]  
    x\_memory[sp] = tr0;

FIG. 10

push ; (1)  
push R0 ; (2)  
push AR0 ; (3)  
put ; (4)  
.....  
pop ; (5)  
pop AR0 ; (6)  
pop R0 ; (7)

FIG. 11A FIG. 11B FIG. 11C FIG. 11D FIG. 11E

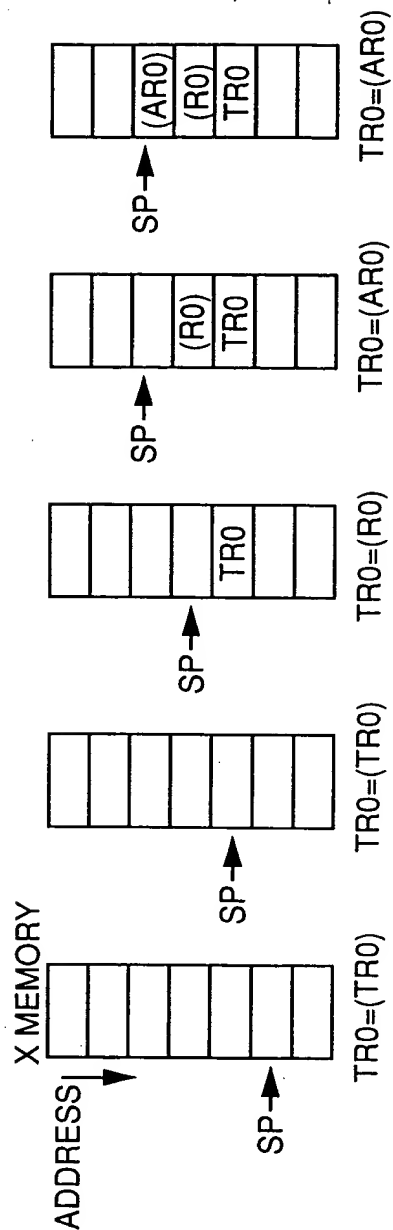


FIG. 11F FIG. 11G FIG. 11H

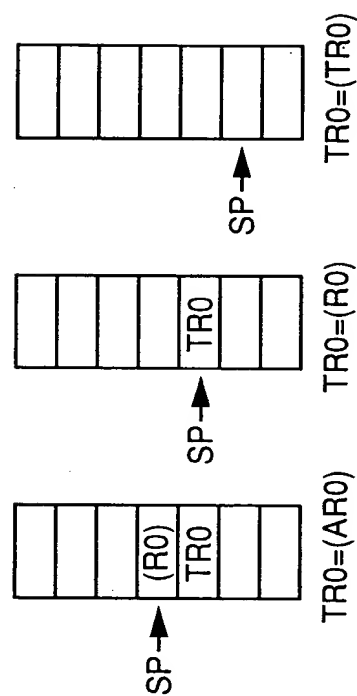
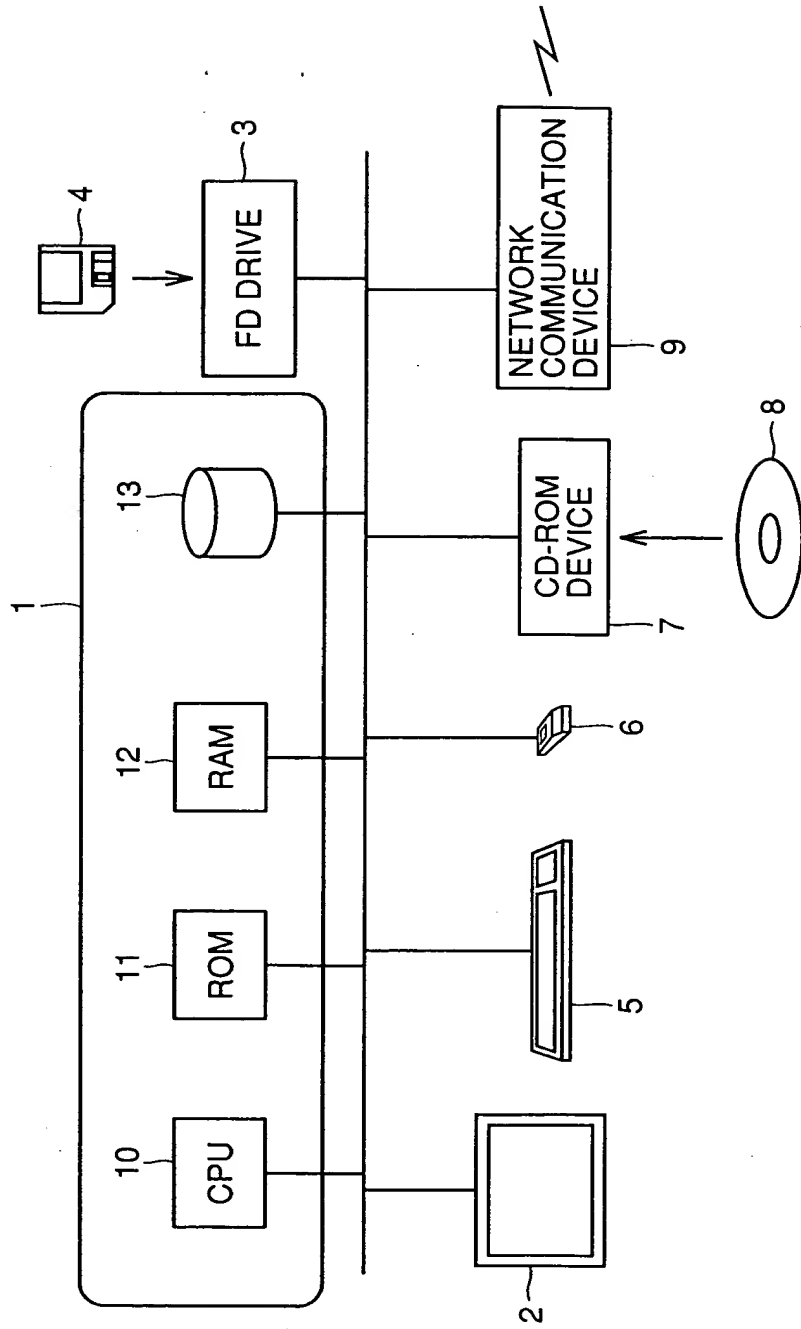




FIG.12



**FIG.13A** { MPUSH R0, AR0; (1)  
 .....  
 MPOP ;(2)

**FIG.13B** { push ;  
 push R0 ;  
 push AR0 ;  
 put ;

**FIG.13C** { pop ;  
 pop AR0 ;  
 pop R0 ;

**FIG.14**

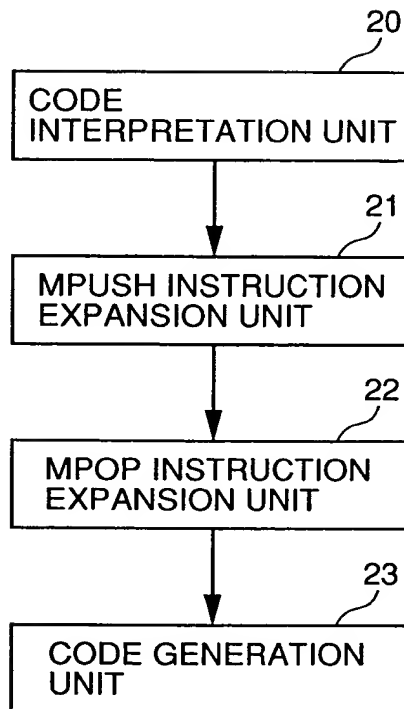


FIG. 15

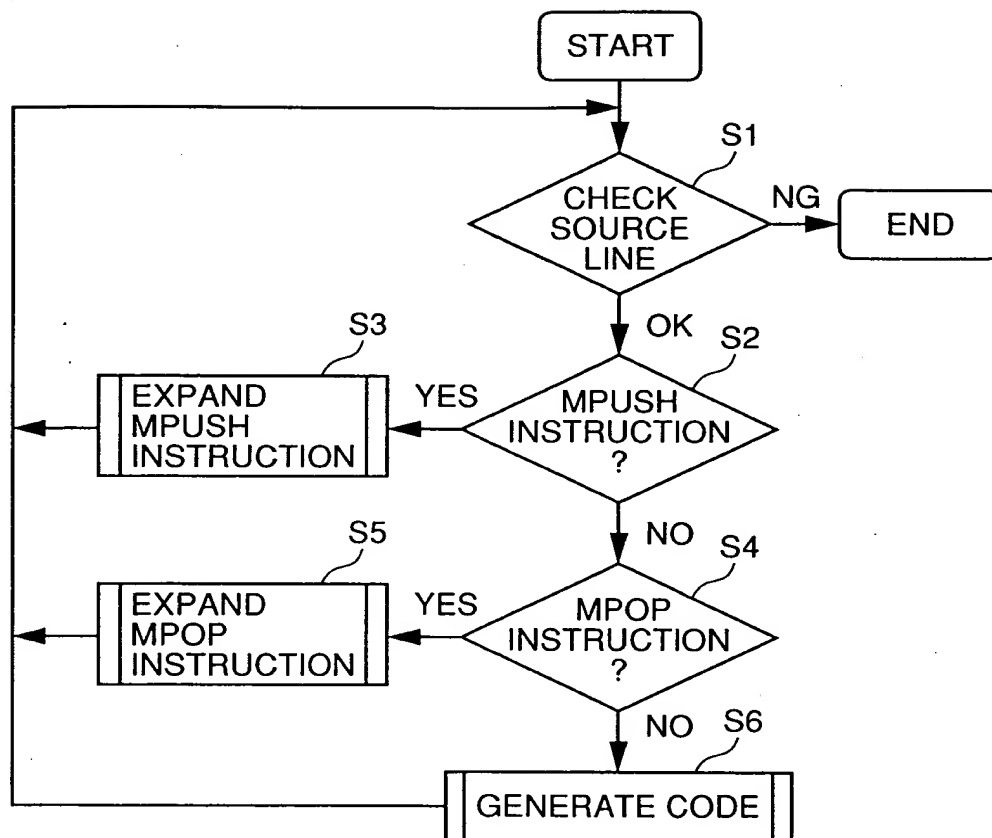


FIG.16

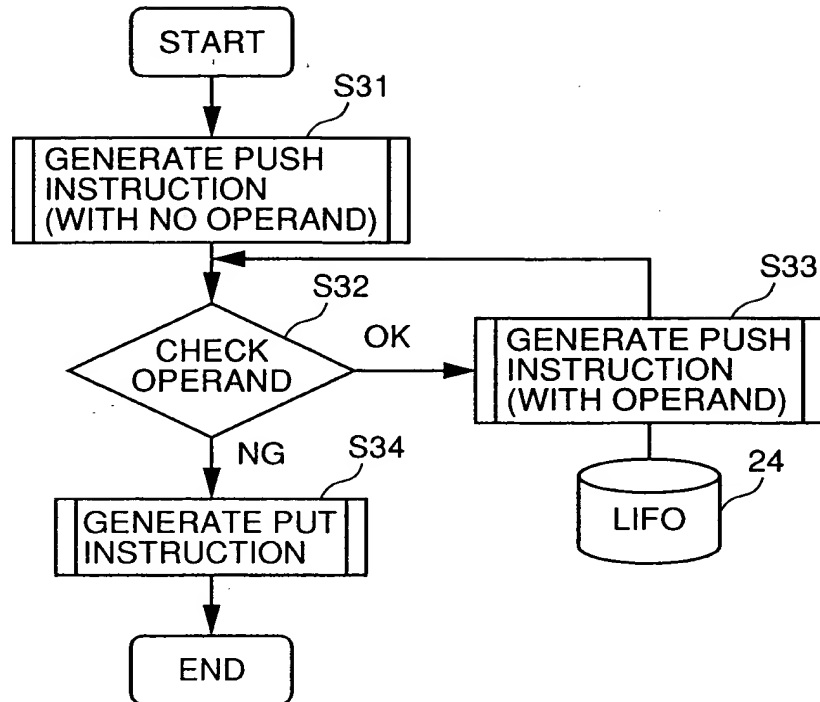


FIG.17

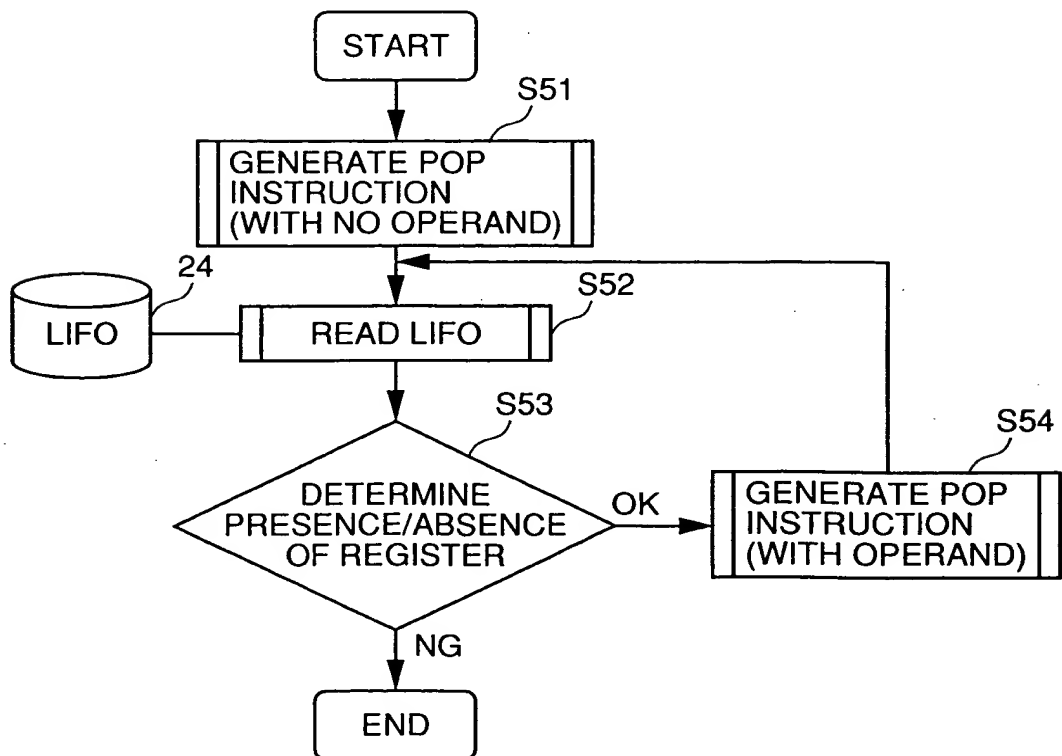


FIG.18A

{ MPUSH TR0, AR0; (1)  
 . . . . .  
 MPOP ;(2)

FIG.18B

{ push ;  
 push AR0 ;  
 put ;

FIG.18C

{ pop ;  
 pop AR0 ;

FIG.19

